NAVAL POSTGRADUATE SCHOOL
Monterey , California

THESIS

IMAGE PROCESSING TECHNIQUES
FOR ACOUSTIC IMAGES

by

Brian P. Murphy

June 1991

Thesis Advisor: Roberto Cristi

REPORT DOCUMENTATION PAGE		Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION / AVAILABILITY OF REPORT
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE		Approved for public release; distribution is unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL (If applicable) EC	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000

8a. NAME OF FUNDING / SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)		10 SOURCE OF FUNDING NUMBERS

		PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

IMAGE PROCESSING TECHNIQUES FOR ACOUSTIC IMAGES

12 PERSONAL AUTHOR(S)
MURPHY, Brian P.

13a TYPE OF REPORT Master's Thesis	13b TIME COVERED FROM _____ TO _____	14 DATE OF REPORT (Year, Month, Day) 1991 June	15 PAGE COUNT 77

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the US Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	image processing, segmentation, edge detection, Kalman filtering, median filtering

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The primary goal of this research is to test the effectiveness of various image processing techniques applied to acoustic images generated in MATLAB. The simulated acoustic images have the same characteristics as those generated by a computer model of a high resolution imaging sonar. Edge Detection and Segmentation are the two image processing techniques discussed in this study. The two methods tested are a modified version of the Kalman filtering and median filtering.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL CRISTI, Roberto	22b TELEPHONE (Include Area Code) 408-646-2223	22c OFFICE SYMBOL EC/Cx

DD Form 1473, JUN 86   *Previous editions are obsolete*   SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

UNCLASSIFIED

Image Processing Techniques for Acoustic Images

by

Brian P. Murphy
Lieutenant, United States Navy
B.S., United States Naval Academy, 1984

Submitted in partial fulfillment of the
required of degree for

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1991

ABSTRACT

The primary goal of this thesis research is to test the effectiveness of various image processing techniques applied to acoustic images generated in MATLAB. The simulated acoustic images have the same characteristics as those generated by a computer model of a high resolution imaging sonar. Edge Detection and Segmentation are the two image processing techniques discussed in this study. The two methods tested are a modified version of Kalman filtering and median filtering.

TABLE OF CONTENTS

I. INTRODUCTION

A. MOTIVATION FOR STUDY

In the study of acoustic imaging, the need arises to classify objects located on or near the sea bottom. The classification process can be related to various areas of interest, such as sea-bottom profiling, mine hunting, undersea navigation and target tracking. This process can become difficult due to the presence of bottom backscatter noise that is generated in the ocean environment. Through implementation of well-known image processing techniques, the image classification process can be made more efficient.

The first step in the classification process requires that the object be separated from the noisy background. This process is called segmentation and it is the first step for image recognition and understanding. Several techniques can be used for image segmentation. They range from ad hoc techniques based on simple thresholding to more sophisticated ones which use statistical models of images. In this thesis, we developed several techniques based on Kalman Filtering and Median Filtering to achieve the desired segmentation of the object. These techniques were used on simulated acoustic images with Gaussian and Rayleigh background noise developed in MATLAB. In addition to segmentation, an algorithm was developed to detect the edges of the simulated acoustic

1

images.   The image processing algorithms were written in MATLAB and processed on the Sun SPARC 1 workstation.

This thesis is organized as follows: the imaging scenario is discussed in Chapter II, a discussion of imaging processing techniques for segmentation and edge detection is presented in Chapter III, and the results of applying these techniques to the simulated acoustic images are described in Chapter IV. Chapter V relates conclusions and recommendations derived from the study.

II. IMAGING BACKGROUND

A. BACKGROUND

In this thesis we consider sonar images generated by a computer simulator. The computer model generates images of a class of targets, either a sphere, a cylinder, or a rectangular bar simulated in the ocean environment. The disturbance primarily affecting the acoustic images in this model are due to bottom backscatter noise.

A high resolution imaging sonar can be modeled as a point source emanating an acoustic plane wave of some frequency typically in the 100 KHz to 2 MHz range. As the plane wave travels through a medium such as the ocean, many factors influence it. These factors include temperature, depth, pressure, salinity, and surface weather conditions. Additionally, as the plane wave approaches the boundaries of the medium other losses occur. These boundaries are created by the different layers of water and sediment that comprise the ocean environment. Each layer has an associated characteristic impedance which affects the transmission of an acoustic plane wave traveling through it. This difference in impedance generally tends to alter the direction of wave propagation. [Ref. 2]

As the acoustic plane wave strikes the target, part of the energy is reflected back toward the sonar, while the remaining

3

energy passes the target and strikes the ocean bottom.
Depending on the type of bottom, some of the incident energy
will be reflected, while the remaining energy is transmitted
into the bottom sediment layer. The energy transmitted within
the bottom sediment layer is further transmitted and reflected
depending on the characteristic impedance of the local
material present. The local characteristic impedance can vary
depending on whether the bottom is composed of mud, mud and
sand, or sand and rock. Eventually, the energy reflected
within the bottom layer returns to the ocean layer to combine
with the energy reflected directly at the ocean bottom
interface. This results in bottom backscatter noise.

B. ACOUSTIC IMAGES

The acoustic images generated from the computer model used
in this thesis can be a sphere, a cylinder or a rectangular
bar. The model first creates a two-dimensional perspective
view of the target based upon programmable parameters defined
by the user. For example, the parameters include target type,
target size, target range, target height, sonar height, and
viewing angle. The perspective images do not contain the
presence of bottom backscatter noise.

After generating the perspective view of the image, the
next step in the imaging process involves combining the
visible target voxels from the perspective view of the image
with programmable sonar system parameters and sea-bottom
backscatter characteristics. Figure 1 presents a diagram of

4
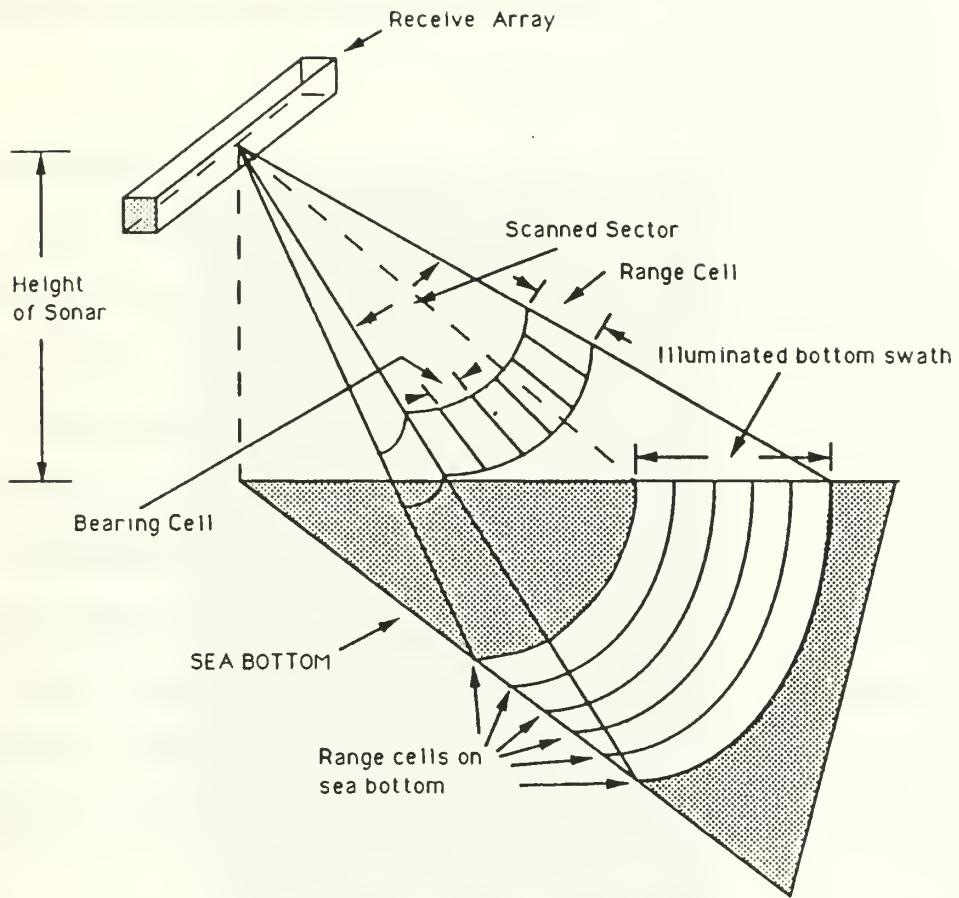
Figure 1. Diagram representing bottom backscatter model
[from Ref. 1]

the bottom backscatter model [Ref. 1]. The bottom backscatter characteristics can be random uniform noise or random Rayleigh noise. Figure 2 illustrates an example of an acoustic image surrounded by random Rayleigh bottom backscatter noise.

C. SIMULATED ACOUSTIC IMAGES

Due to difficulties with processing and displaying the data generated from the high resolution imaging sonar model, simulated acoustic images with random Gaussian and Rayleigh background noise were developed using MATLAB. These programs generated a target (i.e., a circle), surrounded by either a Gaussian or Rayleigh noise background. The simulated acoustic image subroutines had several programmable parameters such as radius of circle, size of matrix, and value of the variance of the background noise. The subroutines are listed in the appendix. Examples of the simulated acoustic images with variable values of the noise variance are shown in Figures 3 - 8.
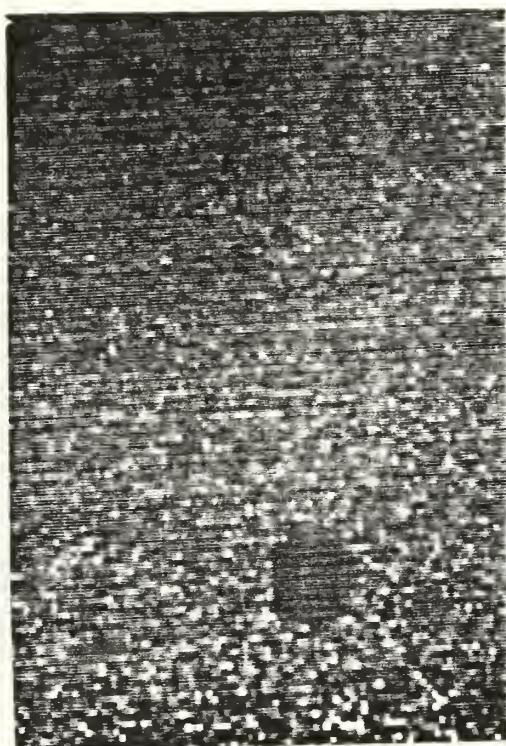
Figure 2. Rectangular bar in random Rayleigh
backscatter noise
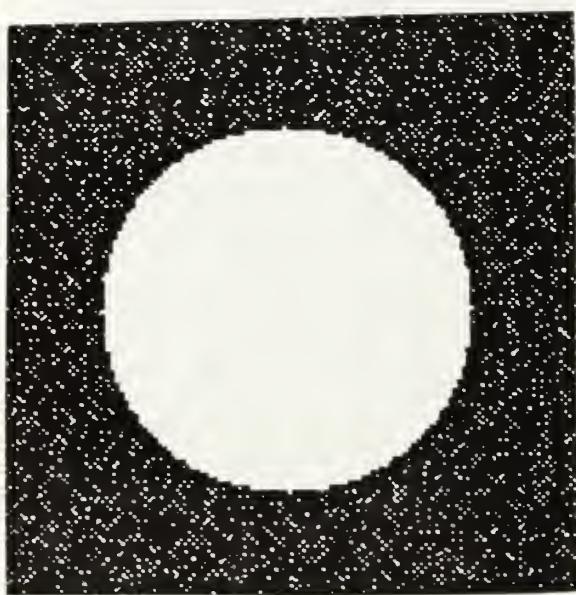
Figure 3. Simulated acoustic image with Gaussian background
noise (Radius = 40, sigma = 0.7)
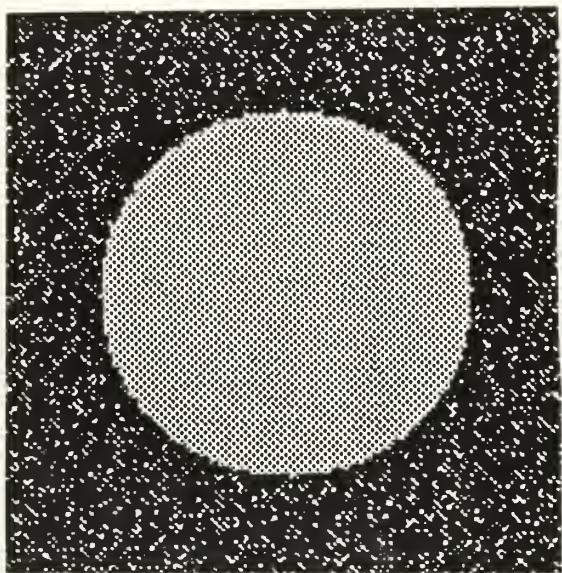
Figure 4. Simulated acoustic image with Gaussian background
noise (Radius = 40, sigma = 1.0)
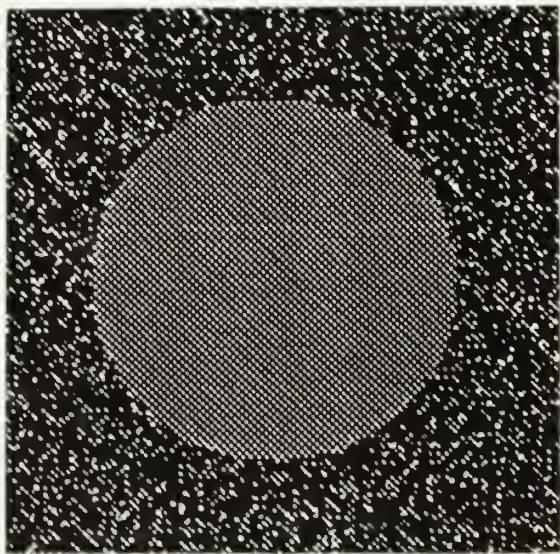
Figure 5. Simulated acoustic image with Gaussian background noise (Radius = 40, sigma = 1.5)
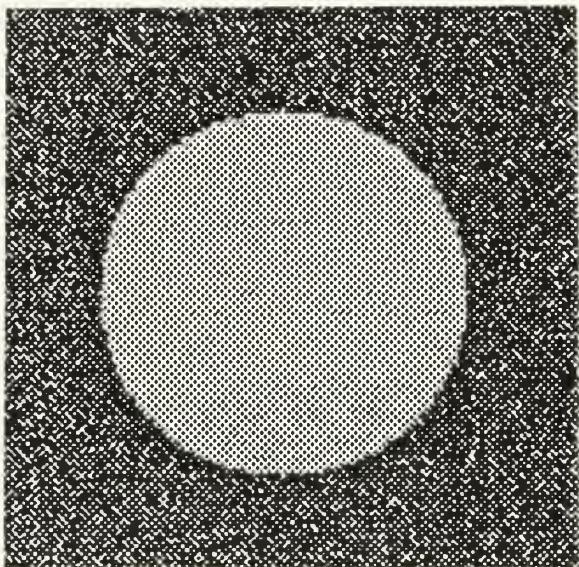
Figure 6. Simulated acoustic image with Rayleigh background
noise (Radius = 4, sigma = 0.7)
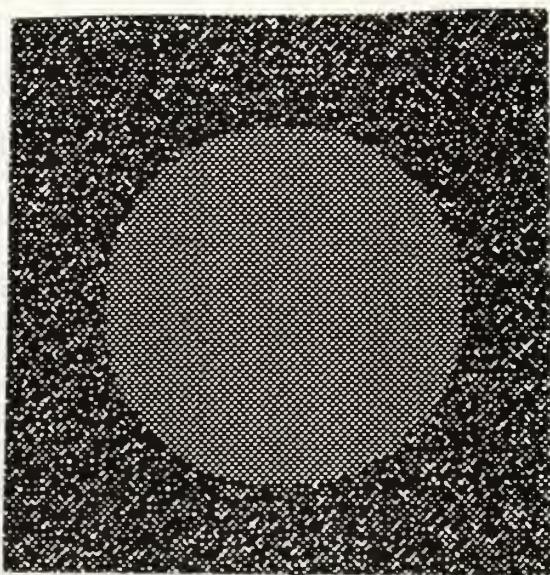
Figure 7. Simulated acoustic image with Rayleigh background
noise (Radius = 4, sigma = 1.0)
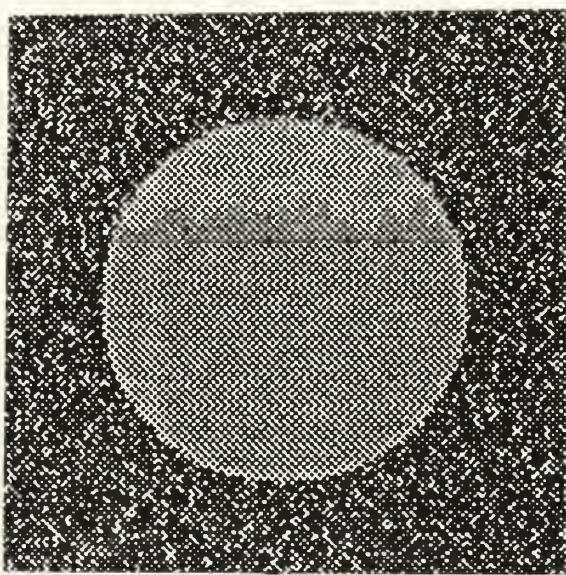
Figure 8. Simulated acoustic image with Rayleigh background
noise (Radius = 40, sigma = 1.5)

III. IMAGE PROCESSING TECHNIQUES

A. GENERAL

This chapter will discuss some of the spatial image processing techniques that were developed for processing the simulated acoustic images in MATLAB. These methods focus primarily on edge detection and segmentation.

An image can be considered as a matrix of light intensity levels that can be manipulated using computer algorithms in MATLAB. Although none of the algorithms developed can be used, as of now, in a real time sense, they provide some insight into the feasibility of imaging processing techniques. In the next section we present an algorithm for edge detection to be applied to images corrupted by external disturbances.

B. EDGE DETECTION

Segmentation is a process that divides an image into separate distinct parts or objects. This is generally the first step in any image processing application because it is at this step that the object of interest is detected from the image for further processing. The process of segmentation is based not only on discontinuities between grey level values within an image matrix, but also on clustering of pixels with similar intensity levels.

An edge can be considered as a boundary between two distinct regions characterized by different levels of

14

intensity. The idea of edge detection is based on recognizing a distinct change in adjacent pixel values. The edge detector for this study will display edges as light pixels and the background as dark pixels.

C.   EDGE DETECTION BY DIFFERENTIATION

Standard techniques of edge detection require the computation of a local derivative operator. Although several choices are available, the most widely used is based on a gradient operator, which can be represented as a two dimensional vector:

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{1}$$

associated with each pixel (x,y) of the image, where f(x,y) represents the pixel intensity level within the image matrix. To determine the location of edges within an image matrix, the magnitude of the vector $G[f(x,y)]$ defined by $|G[f(x,y)]| = [G_x^2 + G_y^2]^{\frac{1}{2}}$ must be computed. [Ref. 3]

Since an image is a matrix defined in a discrete domain, the derivative must be approximated by finite differences. This can be obtained by convolving the intensity pixel data f(x,y) with a mask representing the local operator. In this way, we can compute the two components of the gradient as:

$$G_x(x,y) = \sum_{m,n} h_x(m,n)f(x-m,y-n) \qquad (2)$$

$$G_y(x,y) = \sum_{m,n} h_y(m,n)f(x-m,y-n) \qquad (3)$$

where h_x and h_y represent the impulse responses of the horizontal and vertical filters. In these convolutions, the sums range over the whole field of definition of the image with particular care taken at the boundaries. The operator kernels h_x and h_y in general have a finite region of support as shown in Figure 9.

(-1,1)	(0,1)	(1,1)
(-1,0)	(0,0)	(1,0)
(-1,1)	(0,-1)	(1,-1)

Figure 9. Region of support for the gradient operator

The particular values assumed by h_x and h_y are shown respectively in Figures 10 and 11.

-1	-2	-1
0	0	0
1	2	1

Figure 10.   Operator kernel for h_x

-1	0	1
-1	0	2
-1	0	1

Figure 11.   Operator kernel for h_y

The gradient vector, G_x, is maximum in correspondence of the horizontal edges within the image while the gradient vector, G_y, is maximum at the vertical edges.   The magnitude of the vector is proportional to the edges of the image.

Although it is very simple to implement, the gradient method does not provide any filtering to remove noise and, in general, is not used on noisy images.   This is due to the same reason we do not perform differentiation on a noisy signal. Also, since the gradient method does not provide any estimate

of the intensity levels of the original image, it cannot be used in segmenting noisy images.

Different alternatives which proved to be effective in the presence of noise and external disturbances have been investigated. The whole idea is to try to detect the regions of the image of sufficiently large size to be classified as object or background. Two techniques have been investigated, the median filter and a modified version of the Kalman filter. The combination of these two techniques seem not only to yield satisfactory performance, but they are also feasible for use on a standard microcomputer.

D. A KALMAN FILTER BASED EDGE DETECTION

The particular class of signals we address are represented by piecewise constant data or data slowly changing within compact regions. In this case, we can model each row of data by a state space equation as follows:

$$x(k+1) = \Phi x(k) + v(k) \qquad (4)$$

$$y(k) = Cx(k) + w(k) \qquad (5)$$

where

    x(k) is the true pixel intensity
    y(k) is the measured pixel intensity
    v(k) is a random forcing function
    w(k) is random measurement noise

with initial conditions at the boundary of each region. [Ref. 4]

The matrices Φ and C are determined from the piecewise constant assumption of the data and several models will be

18

used. In this research we will use one of two models, first order or second order. Higher order models would considerably complicate the algorithm without any significant improvement.

1. First Order Case

In this case, we consider the true intensity as a random walk defined by:

$$x(k+1) = x(k) + v(k)$$

(6)

where the true pixel intensity level x(k) is modeled as the output of a first order linear system. In this case, x(k) is a scalar with $\Phi = 1$ and C = 1. The model (Equation 5) is valid within regions and it changes initial conditions at the boundaries of each region.

2. Second Order Case

With this model, we consider the object as having piecewise constant intensity levels similar to a ramp in order to take into account drifts. The model equation is given by:

$$x(k+1) = 2x(k) - x(k-1) + v(k)$$

(7)

where the true intensity level is modeled as the output of a double integrator. In this case, the state x(k) is a two-dimensional vector and the model matrices are given by:

$$\Phi = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

As in the previous case, the model is valid within the regions and is re-initialized at each edge. In both cases, the noise term v(k) defined by its covariance matrix Q, models differences in data within the regions. Also, it will be seen

19

that this matrix can be used in order to prevent the
estimation algorithm from losing sensitivity as k increases.

The Kalman Filter can then be used to estimate the
true pixel intensity. The Kalman Filter equations for pixel
intensity estimation are defined as:

$$\hat{x}(k+1/k) = \Phi\hat{x}(k/k) \tag{8}$$

$$\hat{y}(k+1/k) = \Phi\hat{x}(k+1/k) \tag{9}$$

$$\hat{x}(k+1/k+1) = \hat{x}(k+1/k) + G(k+1)[y(k+1) - \hat{y}(k+1/k)]^2 \tag{10}$$

where

$\hat{x}(k+1/k)$ is the prediction of the true pixel
intensity at x(k + 1) given the data through point k.

$\hat{y}(k+1/k)$ is the predicted measured pixel intensity
at k + 1 given the data up to point time k.

$\hat{x}(k+1/k+1)$ is the prediction of the true pixel
intensity at k + 1 given the data through point k + 1.
The Kalman Filter gain equations are defined as:

$$P(k+1/k) = \Phi P(k/k)\Phi' \tag{11}$$

$$G(k+1) = P(k+1/k)C'\left[CP(k+1/k)C' + \sigma_w^2\right]^{-1} \tag{12}$$

$$P(k+1/k+1) = [I - G(k+1)C]P(k+1/k) \tag{13}$$

From the state space model, the random forcing
function v(k) and the random measurement noise w(k) are
assumed to be Gaussian random variables. It is a well-known
property of Kalman Filters that given a set of observations
y(0),...,y(k - 1) of pixel intensity values, we can compute

20

the probability that the next observation belongs to the same
model by using the following relation:

$$P\big(y(k)\,/\,y(k-1),\ldots,y(0)\big) = \frac{1}{\sqrt{2\pi}\sqrt{CP(k)C'+R}}\exp\left[-\frac{1}{2}\frac{\big(y(k)-C\hat{x}(k)\big)^2}{CP(k)C'+R}\right] \quad (14)$$

where the dependency on y(k - 1),...,y(0) is contained in $\hat{x}(k)$
[Ref. 5]. This assumption will hold as long as the data y(k),
belongs to the same model as y(k - 1), y(k - 2), etc. This
will result in the data also having a Gaussian distribution
with average $C\hat{x}(k)$ and covariance $CP(k)C'+R$. Therefore, if we
normalize the error to obtain

$$E(k) = \frac{y(k)-C\hat{x}(k)}{\sqrt{CP(k)C'+R}} \qquad (15)$$

in the ideal case, within a region, the variable E(k) is a
Gaussian random variable with zero mean and covariance equal
to one.

a. Edge Detection

The considerations of the Kalman Filter can be
used to construct an edge detector which is robust in the
presence of noise. From the above considerations, we can
detect the edges of an object within an image matrix by
checking

$$|E(k)| > \text{Threshold } H_1$$

$$|E(k)| < \text{Threshold } H_0$$

where H_1 represents detection of an edge and H_0 represents
detection of no edge. The threshold is determined from the
statistical properties of E(k) and fine tuned by trial and
error.

21

Since E(k) is distributed as a Gaussian random variable with zero mean and covariance equal to one, most of its values are within the interval (3,-3). A reasonable choice of a threshold would be between two and three. Any value of $|E(k)|$ above the threshold means that the measurement of y(k) does not belong to the same model as y(k - 1), y(k - 2), etc., and therefore, results in an edge at time k. Clearly, the higher the threshold, the more likely the possibility of missing an edge; similarly the lower the threshold, the higher the possibility of detecting a false edge. The best choice is a compromise which is a function of the signal-to-noise ratio of the data. When the edge is detected, the algorithm reinitializes the covariance matrix P(k) of the Kalman Filter. Therefore, the general algorithm for edge detection is defined as follows [Ref. 6]:

- loop

- compute E(k) from Equation (15)
- if $|E(k)| > T$ then an edge is detected

  o re-initializes Kalman Filter
- Else

  o no edge detected

  o update Kalman Filter

- go to loop

It can be seen that the algorithm also provides for an estimate of the intensity level $C\hat{x}(k)$ within each of

22

the regions. Due to the filtering properties of the Kalman
Filter, we expect $C\hat{x}(k)$ to be smooth within the region and
the reinitialization process at each edge detected prevents
the algorithm from smoothing across the edges.

b. Segmentation

We can segment a noisy image through
implementation of a Kalman Filter that estimates the true
pixel intensity row by row and column by column. The true
pixel intensity can be estimated by testing the likelihood of
each one of the two hypotheses with respective probabilities:

$$H_0 : P\big(y(k) / y(k-1),\ldots,y(0),\ \text{no edge at } k\big)$$

$$H_1 : P\big(y(k) / y(k-1),\ldots,y(0),\ \text{edge at } k\big)$$

Each one of the two probabilities can be computed by running
two Kalman Filter estimates at each point k as

$$\hat{x}_0(k) = \hat{x}(k-1) + K_0(k-1)\big(y(k-1) - C\hat{x}(k-1)\big) \tag{16}$$

$$\hat{x}_1(k) = \hat{x}(k-1) + K_1(k-1)\big(y(k-1) - C\hat{x}(k-1)\big) \tag{17}$$

with $K_i(k - 1)$ being determined from the standard Kalman Gain
equations:

$$P_i(k-1/k-2) = \Phi P_i(k-2/k-2)\Phi' \tag{18}$$

$$K_i(k-1) = P_i(k-1/k-2)C'\big[CP_i(k-1/k-2)C' + R\big]^{-1} \tag{19}$$

$$P_i(k-1/k-1) = \big[I - K_i(k-1)C\big]P_i(k-1/k-2) \tag{20}$$

As a consequence, we update x(k) and P(k) with either $\hat{x}_0(k)$
and $P_0(k)$ or $\hat{x}_1(k)$ and $P_1(k)$ according to which one of the two
probabilities for H_0 or H_1 is larger. Using this comparison
will yield an image with noise removed.

E. MEDIAN FILTERING

This is a non-linear technique that not only filters out spurious noise from the edges of the image, but also preserves the edges of the image. Median Filtering operates on the principle of replacing the grey level value of each pixel with median grey level value of its neighbors. Recall that the median m of a set of values is such that half the values in the set are smaller than m and half are greater than m.

In general, the median filter can be defined as

$$x(i,j) = \text{median } y(m,n) \qquad (m,n) \in \eta_{ij} \qquad (21)$$

where η_{ij} is a neighborhood of pixel (i,j). We use the nearest neighbors shown in Figure 12, with the center pixel (x_5) being pixel (i,j).

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9
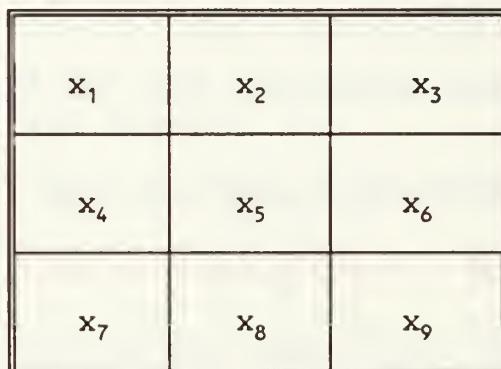
Figure 12. Median filter 3 x 3 mask

In the next section, we see the results of applying the techniques described above.

IV. APPLICATION OF IMAGE PROCESSING TECHNIQUES

A. GENERAL

The techniques discussed in Chapter III were applied to the simulated acoustic images. The algorithms developed were first used to process the simulated acoustic image with random Gaussian background noise. The results obtained from these simulations were analyzed for effectiveness and improvements were made for further application to the simulated acoustic image with Rayleigh background noise. The Sun SPARC 1 workstation and MATLAB software were used to process the image data.

B. SIMULATED ACOUSTIC IMAGE WITH GAUSSIAN BACKGROUND NOISE

The simulated acoustic image used in the study were circles of arbitrary radius with random Gaussian background noise of three different levels. The different background noise levels simulated the various types of bottom backscatter that are present in sonar imaging applications. The simulated acoustic images were previously shown in Figures 3 - 5, respectively.

1. Edge Detection

The modified Kalman Filter algorithm with a first order model performed well in cases where the noise standard deviation is less than one, compared with a difference of three units between object and background. When the noise

increased, the edge detection algorithm had difficulty distinguishing between random background noise and the actual edge of the image. The threshold value, E(k), was adjusted to various levels. If we recall from Chapter II, section D, the error e was compared with a threshold which, in general, ranged from one to three. The reason for these values is the fact that the error signal we use to check for the edge is normalized by its own expected standard deviation. This leads to a random variable which has zero mean, and standard deviation of one, and we know that most of the values are within -3 and 3. In our experiments, the best results obtained occurred with E(k) equal to 2.0. For values of E(k) larger than two, the increased threshold did reduce the noise spikes but the resulting image edges were distorted. The detected edges of the test image for various noise levels are shown in Figures 13 - 15.

The edge detection algorithm based on the second order model, performed far better than that of the first order model. Through trial and error, the threshold value, E(k), equal to 2.0 was found to provide the best results. For sigma equal to 0.7, the algorithm identified the edges of the image perfectly as shown in Figure 16. As the magnitude of sigma was increased to 1.0, the algorithm identified the edges of the image, but had minor problems with some noise spikes (Figure 17). This was still an improvement over the first order model. A post filter, such as the median filtering
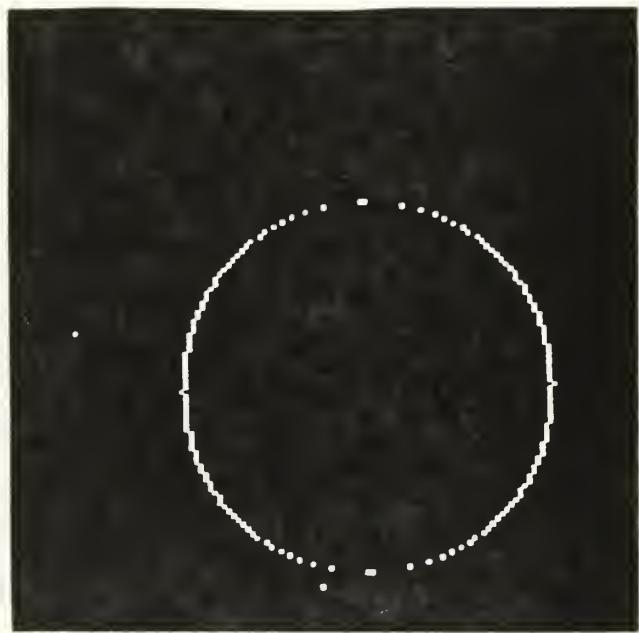
Figure 13. First order model edge detection with Gaussian background noise (E(k) = 2.0, sigma = 0.7)
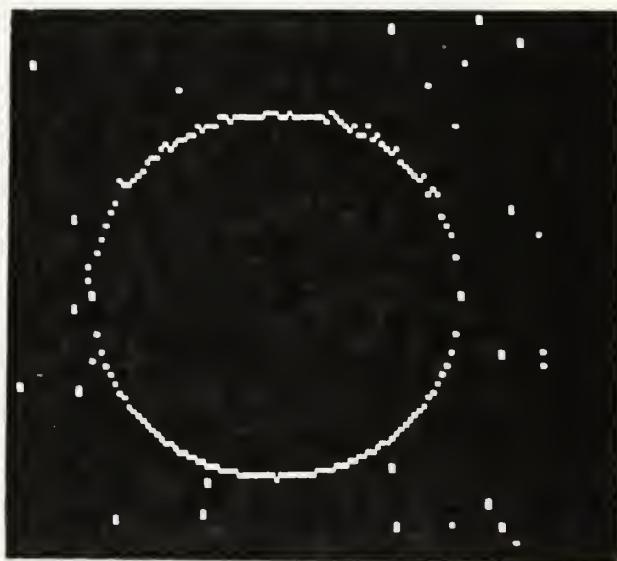
Figure 14.   First order model edge detection with Gaussian
background noise (E(k) = 2.0, sigma = 1.0)
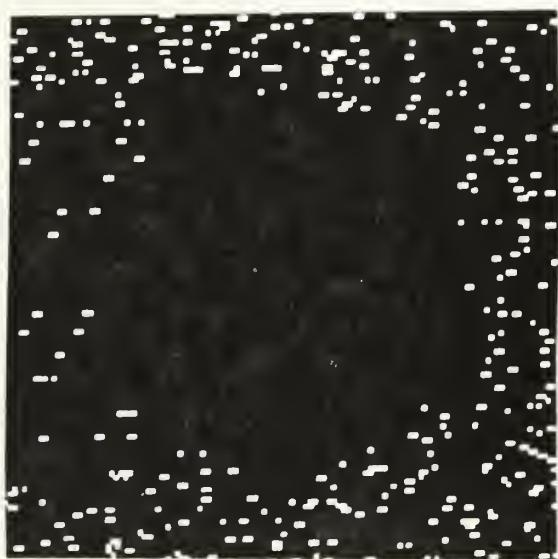
Figure 15. First order model edge detection with Gaussian
background noise (E(k) = 2.0, sigma = 1.5)
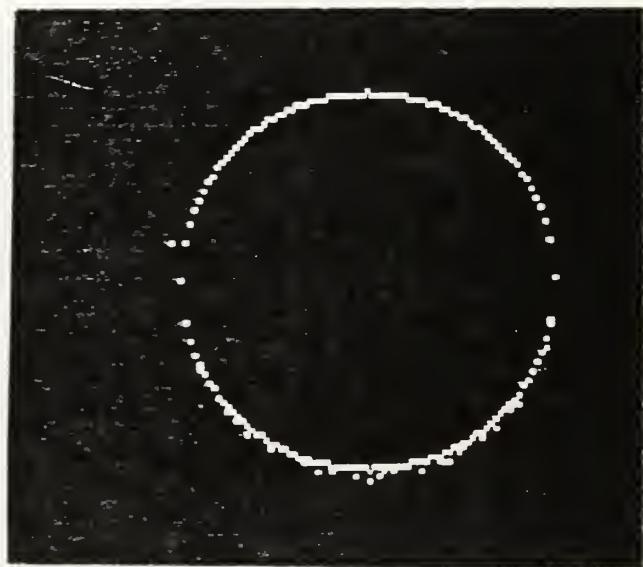
Figure 16. Second order model detection with Gaussian background noise (E(k) = 2.0, sigma = 0.7)
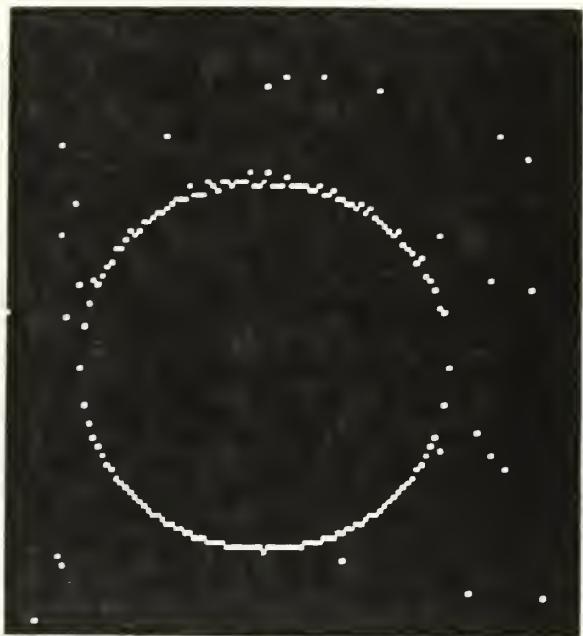
Figure 17. Second order model edge detection with Gaussian background noise (E(k) = 2.0, sigma = 1.0)

algorithm, can be used to remove the noise spikes. For sigma with magnitude equal to 1.5, the second order model failed to identify the true edges of the image (Figure 18). The variation of the threshold value, E(k), did not provide any better results.

2. Segmentation

Using the modified Kalman Filter to estimate the true pixel intensity, provided a very efficient means to segment the object out of the noisy background. For the algorithm based on the first order model, the likelihood of [P(y(k)/y(k-1)...y(0))] has been modified by the addition of an extra term Beta which represents the priori on the edge. This parameter is related to the likelihood of having an edge at any given location, and it can be used to favor estimates with a multitude of edges (i.e., Beta $\approx$ 0) or smooth images with only a few edges (i.e., Beta >> 0).

For the noise with sigma equal to 0.7, the algorithm produced favorable results with beta equal to one (Figure 19). The image was removed from the noisy background with the presence of a small number of noise spikes. These noise spikes were removed by using the median filtering algorithm as a post filter. As the magnitude of sigma was increased to 1.0, the algorithm produced similar results with the addition of more noise spikes (Figure 20). As mentioned above, the use of the median filtering algorithm removed the noise spikes. The algorithm could not function satisfactorily when the
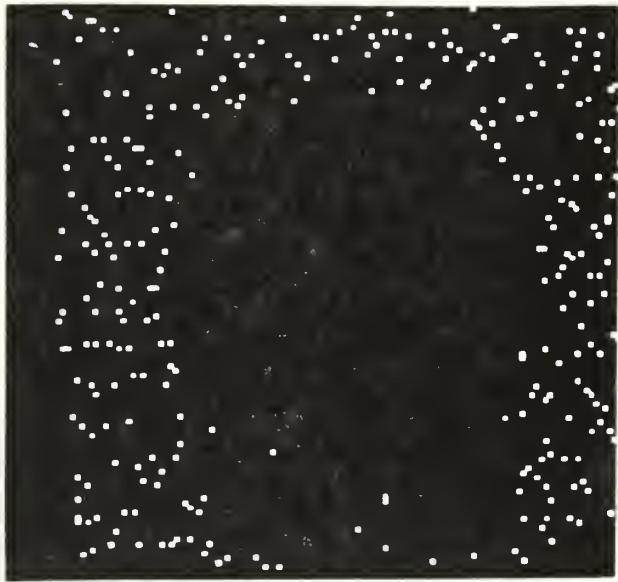
Figure 18. Second order model edge detection with Gaussian
background noise (E(k) = 2.0, sigma = 1.5)
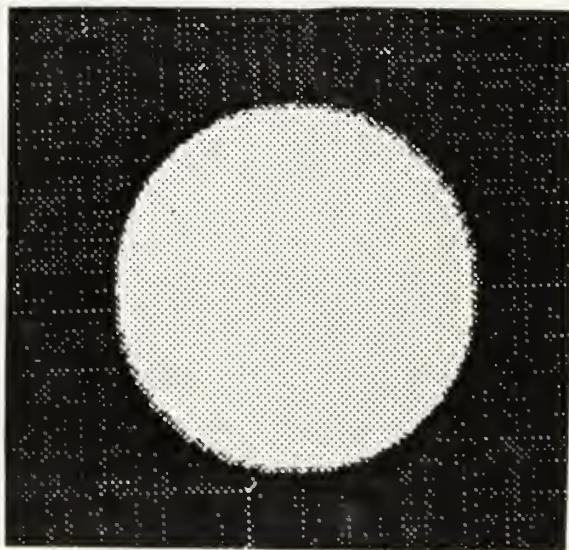
Figure 19.   First order model segmentation with Gaussian
background noise   (Beta = 1.0, sigma = 0.7)
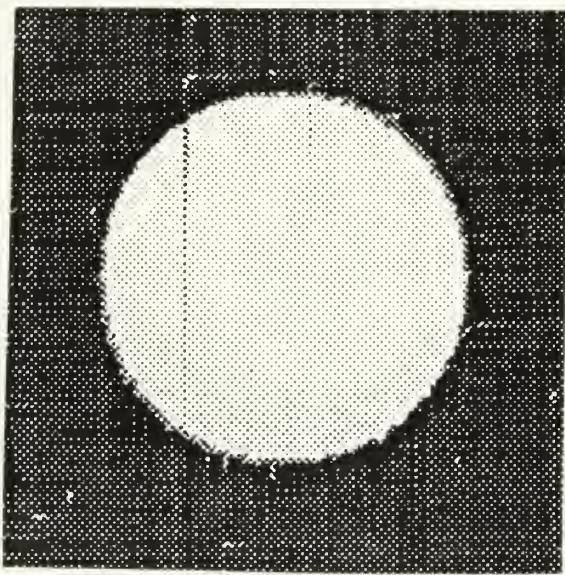
Figure 20.   First order model segmentation with Gaussian
background noise (Beta = 2.0, sigma = 1.0)

magnitude of sigma was increased to 1.5. The value of beta was varied with no appreciable improvement to the algorithm.

The modified Kalman Filter algorithm based on the second order model provided excellent results for the test image with sigma values of 0.7 and 1.0. Figure 21 illustrates the results with sigma equal to one. Using the second order model provided the Kalman Filter an increased sensitivity to differentiate between true pixel intensities and random background noise. It turned out that satisfactory results were obtained with Beta ~ 0. The algorithm produced acceptable results even with a sigma value of 1.5 (Figure 22), although the edges of the image were somewhat distorted. These problems could be resolved through use of a median filter as described in the first order model analysis.

3. Median Filtering

The median filtering algorithm was applied to the simulated acoustic images as previously shown in Figures 3 - 5. The results of this application were as expected (Figures 23 - 25). The algorithm removed single noise spikes and preserved the edges of the images. Median filtering used independently will not produce the desired segmentation of the object from the noisy background. The application of median filtering in combination with the first order model modified Kalman Filter as a post filter, produced acceptable results.
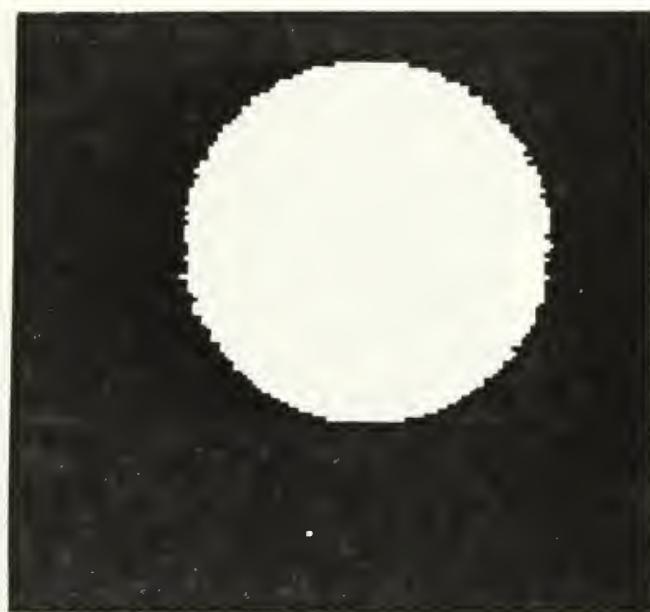
Figure 21.   Second order model segmentation
   (Beta = 0, sigma = 1.0)
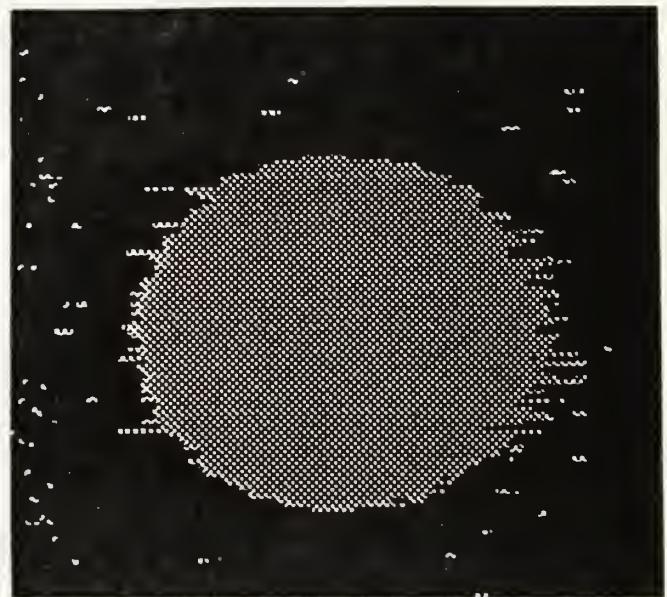
Figure 22. Second order model segmentation
(Beta = 0, sigma = 1.5)

Figure 23. Result of median filtering
(Radius = 40, sigma = 0.7)
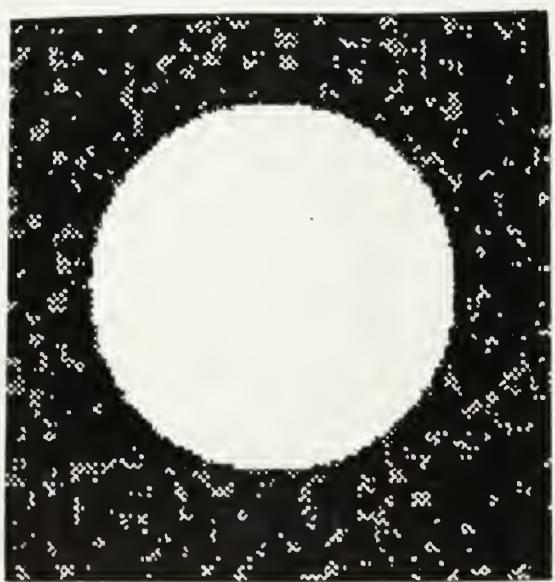
Figure 24.   Result of median filtering
(Radius = 40, sigma = 1.0)
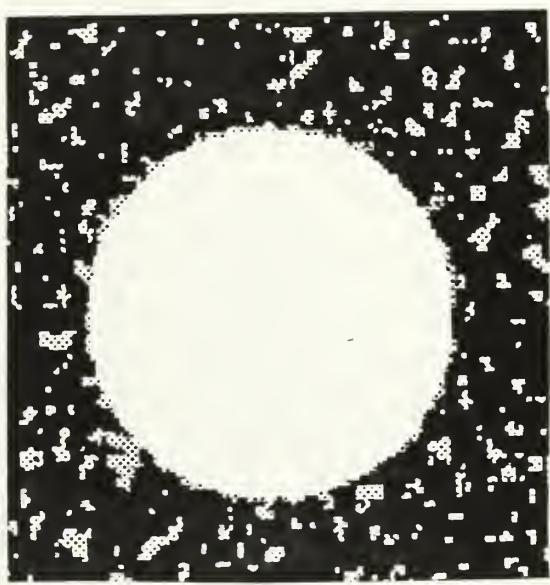
Figure 25. Result of median filtering
(Radius = 40, sigma = 1.5)

C. SIMULATED ACOUSTIC IMAGE WITH RAYLEIGH BACKGROUND NOISE

The simulated acoustic image used in this part of the study was a circle with random Rayleigh background noise. This image was developed in MATLAB to resemble very closely the image generated from the high resolution imaging sonar model (Figure 3).

1. Edge Detection

For values of noise standard deviation less than one, the modified Kalman filter edge detection algorithm has shown acceptable results. The first order model was able to detect the edges of the image but failed to differentiate many noise spikes (Figure 26). The second order presented a far superior result as shown in Figure 27. As the standard deviation of the Rayleigh background noise was increased to one, the modified Kalman filter was still able to produce acceptable results as shown in Figures 28 and 29, respectively. The algorithm had major problems for noise standard deviation values much greater than one (Figures 30 and 31). This was true for both first order and second order models.

2. Segmentation

The modified Kalman filter based on the first order model provided good results. For background noise with a variance equal to 0.7 and Beta equal to 2.0, the algorithm was able to remove the circle from the noisy background (Figure 32). After increasing the variance of the background noise to one, the algorithm was still able to segment the circle from

42
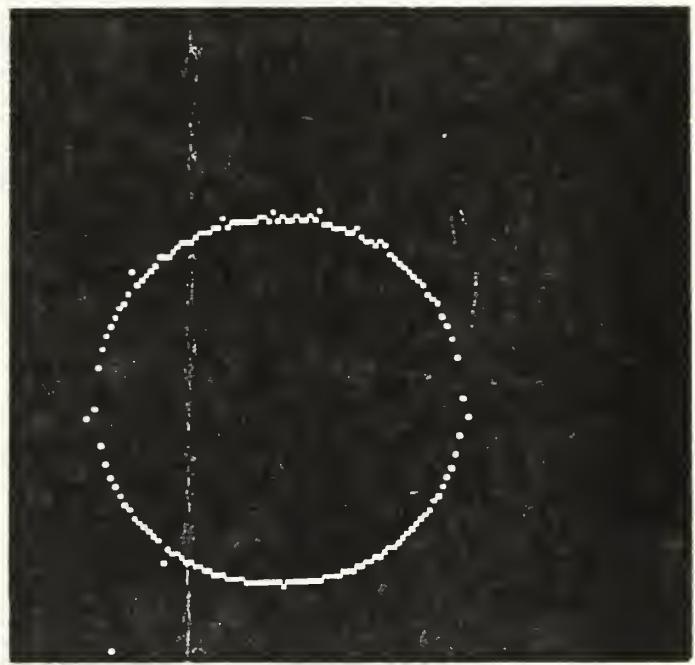
Figure 26. First order model edge detection with Rayleigh
background noise (E(k) = 2.0, sigma = 0.7)
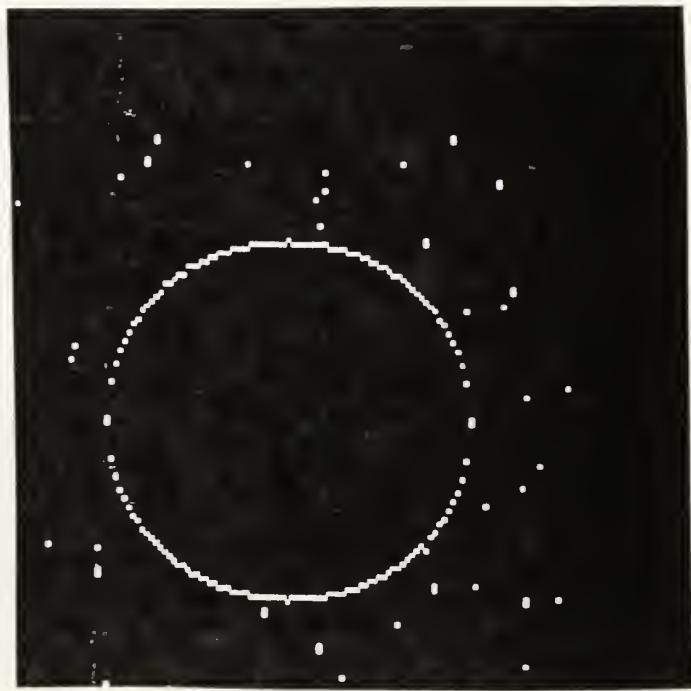
Figure 27. Second order model edge detection with Rayleigh background noise (E(k) = 2.0, sigma = 0.7)
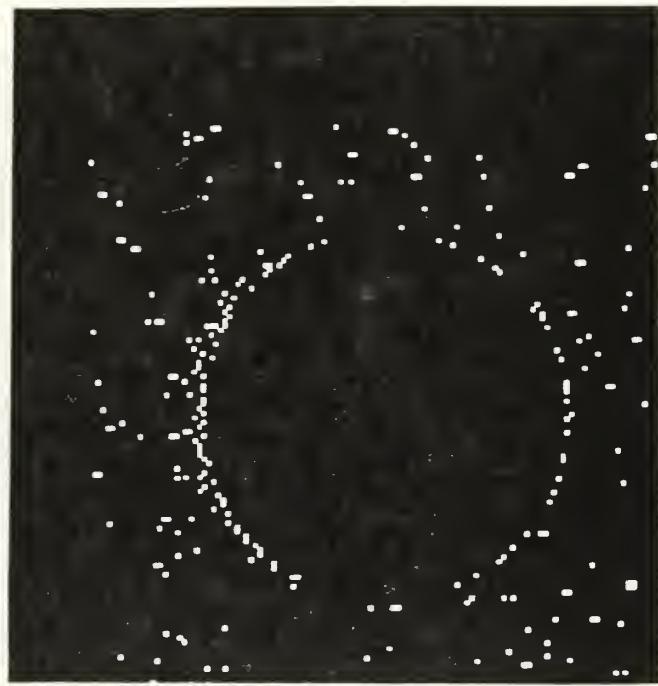
Figure 28. First order model edge detection with Rayleigh background noise (E(k) = 2.0, Sigma = 1.0)
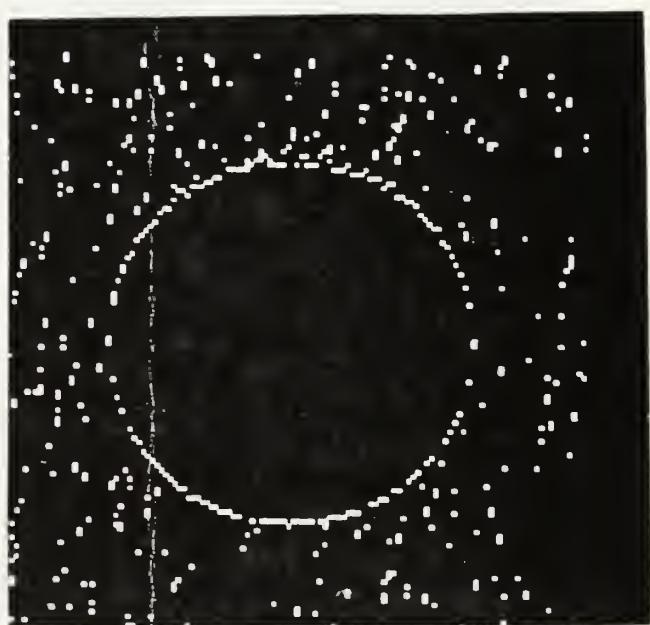
Figure 29. Second order model edge detection with Rayleigh
background noise (E(k) = 2.0, sigma = 1.0)
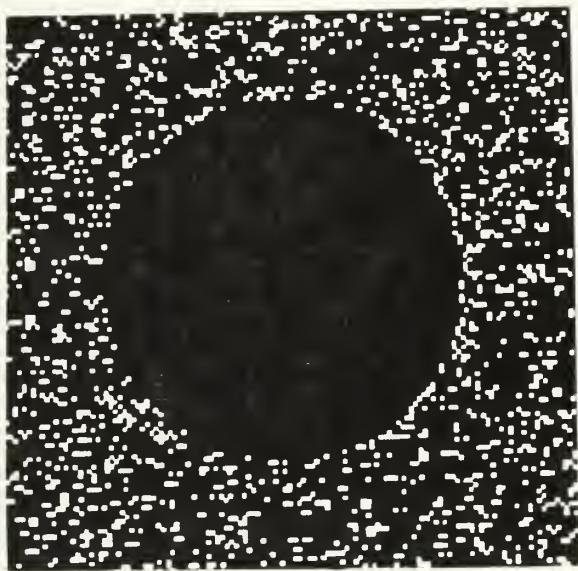
Figure 30. First order model edge detection with Rayleigh
background noise (E(k) = 2.0, sigma = 1.5)
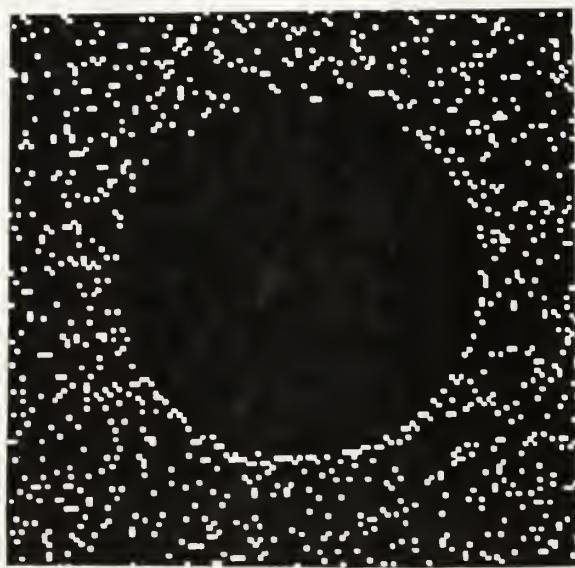
Figure 31. Second order model edge detection with Rayleigh
background noise (E(k) = 2.0, sigma = 1.5)
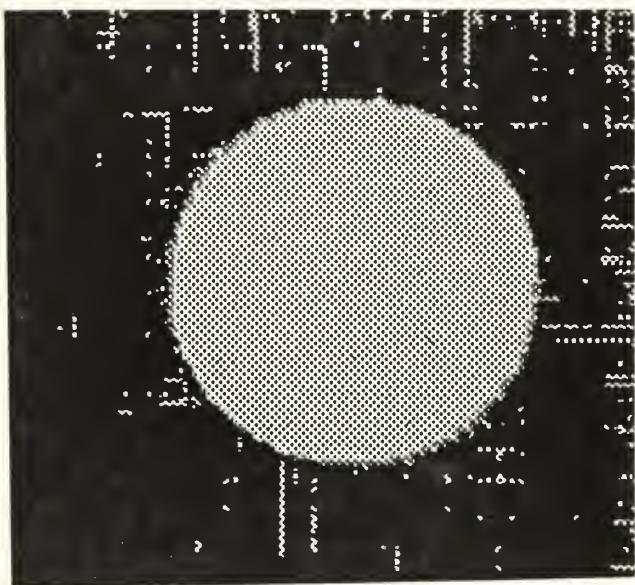
Figure 32.   First order model segmentation with Rayleigh
background noise (Beta = 2.0, sigma = 0.7)

the noisy background (Figure 33). The algorithm was unable to function as the variance of the noise level was increased to 1.5 (Figure 34).

The Kalman filter based on the second order model, provided far better results as expected. For the noise with sigma equal to 0.7, the algorithm completely segmented the simulated acoustic image (Figure 35). Once again the second order model proved to be far more sensitive. As the magnitude of sigma was increased to one, the algorithm still provided good results with some minor noise spikes (Figure 36).
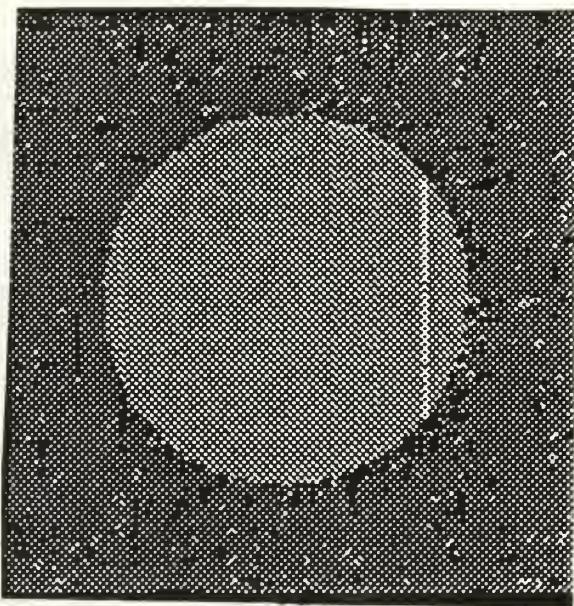
Figure 33.   First order model segmentation with Rayleigh background noise (Beta = 1.0, sigma = 1.0)
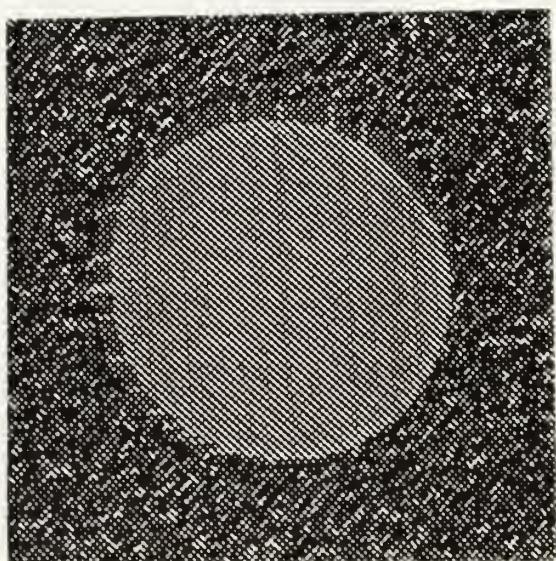
Figure 34. First order model segmentation with Rayleigh background noise (Beta = 2.75, sigma = 1.5)
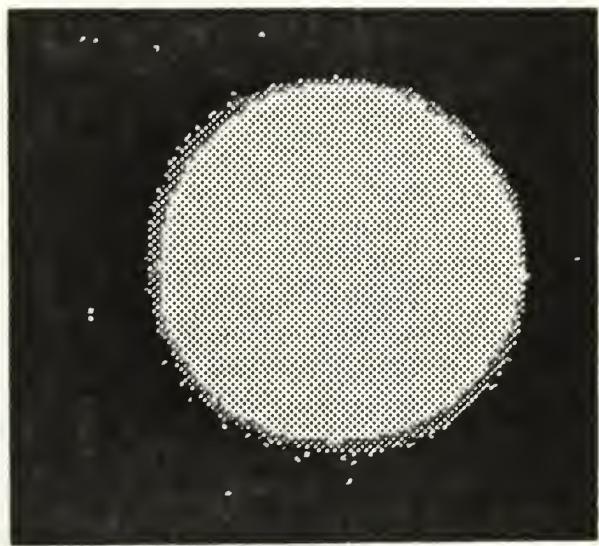
Figure 35.   Second order model segmentation with Rayleigh
background noise (Beta = 0, sigma = 0.7)
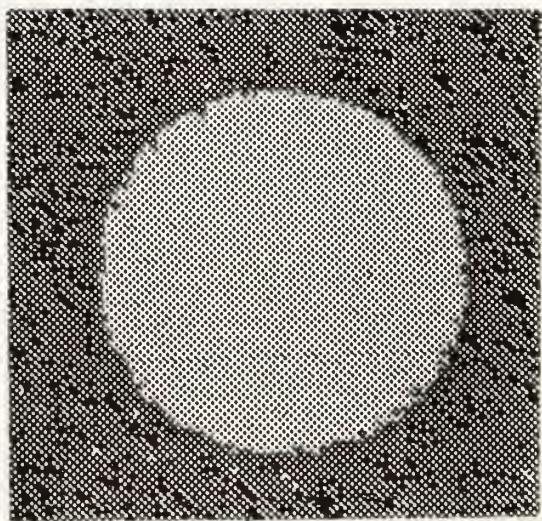
Figure 36.   Second order model segmentation with Rayleigh
background noise (Beta = 0, sigma = 1.0)

3. Median Filtering

The median filtering algorithm has been applied to the simulated acoustic images with random Rayleigh background noise (Figure 6 - 8). The results of the median filtering algorithm are shown in Figures 37 - 39. In all three cases, the algorithm removed noise spikes and preserved the edges of the image.
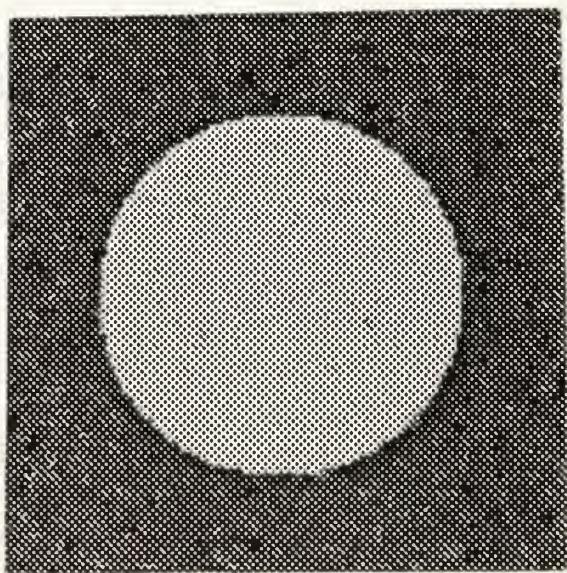
Figure 37.   Median filtering with Rayleigh background noise
(Sigma = 0.7)
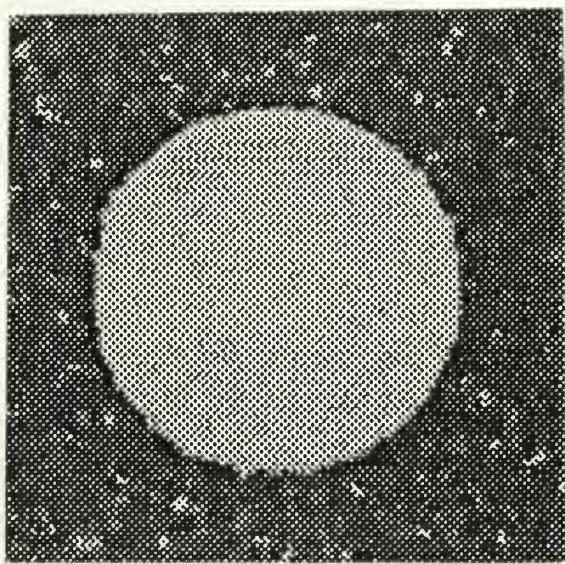
Figure 38.   Median filtering with Rayleigh background noise
(Sigma = 1.0)
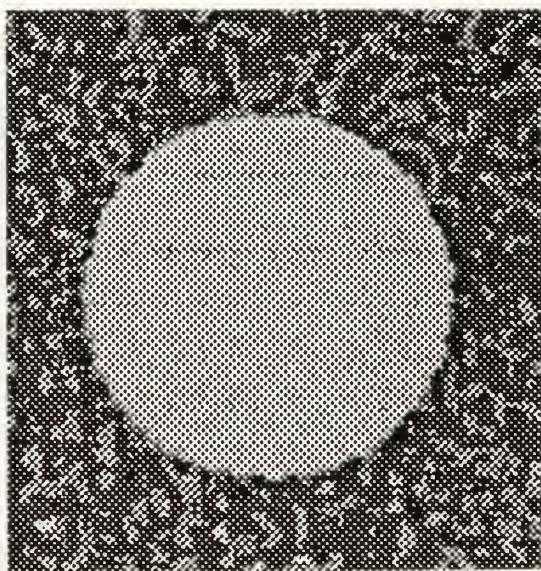
Figure 39. Median filtering with Rayleigh background noise
(Sigma = 1.5)

V. CONCLUSIONS

The image processing algorithms performed well on the simulated acoustic images with random Gaussian and Rayleigh background noise. In both edge detection and segmentation, the modified Kalman filter provided good results in the different types of noise backgrounds.

The results of the edge detection implementation suggests the fact that the modified Kalman filter is robust in its application to Gaussian and Rayleigh background noise. For Gaussian background noise, the optimum threshold value was found to equal 2.0. Using the same threshold value for Rayleigh background noise, the results were very similar to that of the first and second order Gaussian models. As the magnitude of background noise was increased, the edge detection algorithm began to break down. For plausible signal-to-noise ratios, the edge detection algorithm continued to function with acceptable results.

The modified Kalman filter algorithm used for segmentation also provides insight on the robustness of the Kalman filter. In the first order model case, the algorithm had to rely on similar values of a parameter Beta which expresses the likelihood of finding an edge in order to achieve segmentation of the object. In the case of Gaussian background noise, the optimal value for Beta was equal to 2.0. The optimal value

for Beta was also equal to 2.0 for Rayleigh background noise. The best segmentation results were achieved through use of a second order model. In both cases, the optimal value of Beta was equal to zero for Gaussian and Rayleigh background noise.

The concept of median filtering was also introduced in this thesis. The use of median filtering in combination with the modified Kalman filter provided good results. The median filtering algorithm can be implemented as a post filter to further remove spurious noise spikes and preserve the edges of the object.

IMAGE PROCESSING ALGORITHMS

```
***********************************************************
* This is a subroutine that will generate a simulated     *
* acoustic image with Gaussian backround noise. The inputs *
* to the subroutine are circle radius,noise variance and  *
* and matrix size.                                         *
***********************************************************
clear
clg
% Define random Gaussian backround noise
rand('normal');
% Define conditions for simulated acoustic image
N=input('Input the number of row pixels             ');
M=input('Input the number of column pixels          ');
R=input('Input the desired radius of the circle      ');
S=input('Input the value for sigma                    ');
aa=S^2*fix(abs(rand(N,M)));
c=zeros(1,4);
% Define center of image
cx=N/2;
cy=M/2;
% Generate simulated acoustic image
    for i=1:N
        for j=1:M
            x=(i-cx)^2+(j-cy)^2;
                if x<= R^2
                    aa(i,j)=3;
                end
        end
    end
***********************************************************
*   This is a subroutine that will generate a simulated    *
*   acoustic image with random Rayleigh backround noise.   *
*   The inputs to the subroutine are circle radius,noise   *
*   variance and matrix size.                              *
***********************************************************
%Define initial conditions for simulation
c=zeros(1,4);
N=input('Input the number of row pixels             ');
M=input('Input the number of column pixels          ');
R=input('Input the desired radius of circle          ');
S=input('Input value for sigma                       ');
```

```
X=ones(N,M)-rand(N,M);
Y=-2*S^2*log(X);
aa=sqrt(Y);
% Generate random Rayleigh backround noise
aa=rayleigh(N,M,S);
% Define center of image
cx=N/2;
cy=M/2;
% Generate simulated acoustic image
      for i=1:N
            for j=1:M
                  x=(i-cx)^2+(j-cy)^2;
                  if x <= R^2
                     aa(i,j)=3;
                  end
            end
      end
*****************************************************************
*   This is a subroutine that will perform median filtering *
*   on the simulated acoustic images.  The input to         *
*   the subroutine is a matrix  N x N containing simulated   *
*   acoustic image data. The output is a matrix of image     *
*   data representing the median values of the image.        *
*                                                            *
*****************************************************************
% Read simulated acoustic image into MATLAB
a=aa;
[N,M]=size(a);
% Create output matrix
d=zeros(N,M);
% Perform median filtering on data
      for i=2:N-1
            for j=2:M-1
               w=[a(i-1,j-1:j+1) a(i,j-1:j+1) a(i+1,j-1:j+1)]';
               d(i,j)=median(w);
            end
      end
% Save filtered image
 putim(d,'medtest')
******************************************************************
*   This is a subroutine that will compute the Kalman Gain    *
*   values for the first order model case.  The Kalman Gain   *
*   values are computed along the row of pixels of interest.  *
*                                                        *
*   For this case the variable M indicates the number of rows *
*                                                   *
*   in the image matrix.                                      *
*                      *
*                                                            *
******************************************************************
% Define initial error covariance value
p0=10000;
```

```
% Define initial conditions
I=1;
phi=1;
c=1;
R=S^2;
% Generate Kalman gain values
   for i=1:M
       p0=(phi*p0*phi');
       G(i)=(p0*c')/(c*p0*c'+R);
       p0=(I-G(i)*c)*p0;
       s(i)=sqrt(c*p0*c'+R);
   end
*****************************************************************
*   This is a subroutine that will calculate the Kalman Gain   *
*   values for the second order case.  The Kalman Gain values  *
*   are calculated based on each row of pixels in the image    *
*   file.  The variable M represents the number of rows in the*
*   image file.                                                *
*****************************************************************
% Define initial error covariance matrix
p0=[10000 0;0 10000];
% Define initial conditions
I=[1 0;0 1];
phi=[0 1;-1 2];
c=[0 1];
R=S^2;
G=zeros(2,M);
% Generate Kalman Gain values
   for i=1:M
       p0=(phi*p0*phi');
       G(:,i)=(p0*c')/(c*p0*c'+R);
       p0=(I-G(:,i)*c)*p0;
       s(i)=sqrt(c*p0*c'+R);
   end
*****************************************************************
*   This is a subroutine that will compute the gradient        *
*   operator of the simulated  acoustic images.                *
*                                                              *
*                                                              *
*                                                              *
*****************************************************************
% Read in simulated acoustic image data
a=aa;
N=input('Input the number of row pixels              ');
M=input('Input the number of column pixels           ');
% Define output matrix
f=zeros(N,M);
% Create row mask
mx=[-1 -2 -1;0 0 0;1 2 1];
% Create column mask
my=[-1 0 1;-2 0 2;-1 0 1];
x=[mx(1,1) mx(1,2) mx(1,3) mx(2,1) mx(2,2) mx(2,3) mx(3,1) mx(3,2)
```

63

```matlab
mx(3,3)];
y=[my(1,1) my(1,2) my(1,3) my(2,1) my(2,2) my(2,3) my(3,1) my(3,2)
my(3,3)];
% Compute gradient values
   for i=2:N-1
       for j=2:M-1
           ax=[a(i-1,j-1:j+1) a(i,j-1:j+1) a(i-1,j-1:j+1)];
           Gx=conv(ax,x);
           ay=[a(i-1,j-1:j+1) a(i,j-1:j+1) a(i+1,j-1:j+1)];
           Gy=conv(ay,y);
           f(i,j)=sqrt(Gx*Gx'+Gy*Gy');
       end
   end
% Save image for display
 putim(f,'testg')
*********************************************************************
*   This subroutine detects the edges of the simulated      *
*   acoustic images for the first  order case.  The des-    *
*   ired tolerance level and variance of backround noise are *
*   input to the program.                                   *
*                                                           *
*********************************************************************
% Read in test data
y=aa;
T=input('Input the desired tolerance level     ');
S=input('Input the value of noise variance sigma    ');
[N,M]=size(y);
% Define output matrix
e=zeros(N,M);
% Define initial conditions
c=1;
xhat=zeros(N,M);
% Generate Kalman Gain values
kal
% Detect edges of image
   for k=1:N
       t=1;
        for l=1:M
           xhat(k,l+1)=xhat(k,l)+G(1,t)*(y(k,l)-xhat(k,l));
           E=abs((y(k,l+1)-xhat(k,l+1))/s(t));
           if  E > T
               t=1;
               e(k,l+1)=255;
           else
               t=t+1;
               e(k,l+1)=0;
           end
       end
   end
% Save image for display
putim(e,'edtest')
*********************************************************************
```

```
*    This is a subroutine that will detect the edges of the    *
*    the simulated acoustic images for the second order        *
*    case.   The desired tolerance and noise variance are      *
*    input to the program.                                     *
*                                                              *
****************************************************************
% Read in test data
y=aa;
T=input('Input desired tolerance level            ');
S=input('Input value of sigma                     ');
[N,M]=size(y);
% Define initial conditions
e=zeros(N,M);
c=[0 1];
xhat=zeros(N,M);
% Calculate Kalman Gain values
kal2
% Detect edges of image
   for k=1:N
        t=1;
         for l=1:M-1
                xhat(k,l+1)=xhat(k,l)+G(1,t)*(y(k,l)-xhat(k,l));
                E=abs((y(k,l+1)-xhat(k,l+1))/s(t));
                if E > T
                    t=1;
                    e(k,L+1)=255;
                else
                    t=t+1;
                    e(k,l+1)=0;
                end
         end
   end
% Save image for display
putim(e,'ed2test')
****************************************************************
*    This is a subroutine that will segment the simulated      *
*    acoustic images for the first order case.  The algorithm  *
*    will estimate the pixel intensity row by row and          *
*    column by column then average the row and column sum      *
*    for the segmented output.                                 *
****************************************************************
% Read in test data
y=aa;
S=input('Input the value for sigma                ');
B=input('Input value for Beta                     ');
% Define initial conditions
z=sqrt(2*pi);
xhat0=zeros(N,M);
xhat1=zeros(N,M);
xhat00=zeros(N,M);
xhat11=zeros(N,M);
xhatr=zeros(N,M);
```

65

```
xhatc=zeros(N,M);
[N,M]=size(y);
% Compute Kalman Gain values
kal
kal2
% Generate row pixel intensity estimates
   for k=1:N
      t=1;
       for l=M-1
        xhat0(k,l+1)=xhatr(k,l)+G(1,t)*(y(k,l)-xhatr(k,l));
        xhat1(k,l+1)=xhatr(k,l)+G(1,1)*(y(k,l)-xhatr(k,l));
        lp0=-0.5*log(z*s(t))-0.5*((y(k,l+1)-xhat0(k,l+1))^2/s(t)^2);
        lp1=-0.5*log(z*s(1))-0.5*((y(k,l+1)-xhat1(k,l+1))^2/s(1)^2);
        if lp0 > lp1-B
           xhatr(k,l+1)=xhat0(k,l+1);
                t=t+1;
        else  xhatr(k,l+1)=xhat1(k,l+1);
                t=1;
        end
       end
   end
% Generate column pixel intensity estimates
for jj=1:M
     t=1;
    for ii=1:N-1
    xhat00(ii+1,jj)=xhatc(ii,jj)+Gc(1,t)*(y(ii,jj)-xhatc(ii,jj));
    xhat11(ii+1,jj)=xhatc(ii,jj)+Gc(1,1)*(y(ii,jj)-xhatc(ii,jj));

lp0c=-0.5*log(z*s(t))-0.5*((y(ii+1,jj)-xhat00(ii+1,jj))^2/sc(t)^2);

lp1c=-0.5*log(z*s(1))-0.5*((y(ii+1,jj)-xhat11(ii+1,jj))^21/sc(1)^
2);
    if lp0c > lp1c - B
         xhatc(ii+1,jj)=xhat00(ii+1,jj);
           t=t+1;
    else xhatc(ii+1,jj)=xhat(ii+1,jj);
           t=1;
    end
   end
 end
% Compute output image matrix
[xhat]=avg([xhatr]+[xhatc]);
% Save output image for display
putim(xhat,'segtest')
*******************************************************************
*   This is a subroutine that will segment the simulated      *
*   acoustic images for the second order case.  This routine *
*   will estimate the pixel intensity row by row and          *
*   column by column then average the sum of the rows and    *
*   columns for the segmented output image.                   *
*                                                             *
*******************************************************************
```

66

```
% Read in test data
y=aa;
% Define initial conditions
c=[ 0 1 ];
z=sqrt(2*pi);
xhat0=zeros(N,M);
xhat1=zeros(N,M);
xhat00=zeros(N,M);
xhat11=zeros(N,M);
xhatr=zeros(N,M);
xhatc=zeros(N,M);
% Input parameters
S=input('Input the value for sigma                    ');
B=input('Input the value for Beta                     ');
[N,M]=size(y);
% Generate row pixel intensity estimates
for k=1:N
    t=1;
    for l=1:M-1
      xhat0(k,l+1)=xhatr(k,l)+G(1,t)*(y(k,l)-xhatr(k,l));
      xhat1(k,l+1)=xhatr(k,l)+G(1,1)*(y(k,l)-xhatr(k,l));
      lp0=-0.5*log(z*s(t))-0.5*((y(k,l+1)-xhat0(k,l+1))^2/s(t)^2);
      lp1=-0.5*log(z*s(1))-0.5*((y(k,l+1)-xhat1(k,l+1))^2/s(1)^2);
      if lp0 > lp1 - B
          xhatr(k,l+1)=xhat0(k,l+1);
          t=t+1;
      else
          xhatr(k,l+1)=xhat(k,l+1);
          t=1;
      end
    end
end
% Generate column pixel intensity estimates
 for jj=1:M
     t=1;
      for ii=1:N-1

xhat00(ii+1,jj)=xhatc(ii,jj)+Gc(1,t)*(y(ii,jj)-xhatc(ii,jj));

xhat11(ii+1,jj)=xhatc(ii,jj)+Gc(1,1)*(y(ii,jj)-xhatc(ii,jj));

lp0c=-0.5*log(z*s(t))-0.5*((y(ii+1,jj)-xhat00(ii+1,jj))^2/sc(t)^2;

lp1c=-0.5*log(z*s(1))-0.5*((y(ii+1,jj)-xhat11(ii+1,jj))^2/sc(1)^2;
         if lp0c > lp1c - B
               xhatc(ii+1,jj)=xhat00(ii+1,jj);
                 t=t+1;
         else
                xhatc(ii+1,jj)=xhat(ii+1,jj);
                 t=1;
           end
  end
```

```
% Compute output image matrix
[xhat]=avg([xhatr]+[xhatc]);
% Save image for output
putim(xhat,'seg2test')
```

LIST OF REFERENCES

1.   Bahl, Rajender and Powers, John P., *Computer Model of a High-Resolution Imaging Sonar*, Technical Report, Naval Postgraduate School, Monterey, CA, 1990.

2.   Burdic, William S., *Underwater Acoustic System Analysis*, Prentice-Hall, Inc., 1984, pp. 126 - 141.

3.   Gonzalez, Rafael C. and Wintz, Paul, *Digital Image Processing*, 2d ed., Addison-Wesley Publishing Co., 1987, pp. 331- 340.

4.   Burl, Jeff, "Derivation of Kalman Filter Equations," course notes, EC 3310, Naval Postgraduate School, Monterey, CA.

5. Cristi, Roberto, "Kalman Filtering: Application to System Validation," course notes, EC 4310, Naval Postgraduate School, Monterey, CA

6. Merritt, Paul, *Video Tracking of Objects on an Enhanced PC System*, Naval Postgraduate School Master's Thesis, December 1989.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center   2
   Cameron Station
   Alexandria, Virginia 22304-6145

2. Library, Code 52   2
   Naval Postgraduate School
   Monterey, California 93943-5002

3. Chairman, Code EC   1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93940-5000

4. Professor Roberto Cristi, EC/Cx   1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93940-5000

5. Professor Jeff Burl, EC/Bu   1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93940-5000

6. Dr. Rajendar Bahl   1
   Center for Applied Research in Electronics
   Indian Institute of Technology
   New Delhi, 110016
   India

7. Professor John Powers, Code EC/Po   1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93940-5000